Report No. 343

COO-1469-0142

DESIGN OF A DISPLAY PROCESSING UNIT IN A MULTI-TERMINAL ENVIRONMENT

by

Raphael Hostovsky

July 30, 1969

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. 343

DESIGN OF A DISPLAY PROCESSING UNIT IN A MULTI-TERMINAL ENVIRONMENT*

by

Raphael Hostovsky

July 1969

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

* This work was supported in part by Contract No. U. S. AEC AT(11-1)1469 and was submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science, July 1969.

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to Dr. C. W. Gear for help and suggestions both in the writing and in the supervision of this thesis.

Mr. Martin Michel and Mr. Robert Miller also deserve thanks for their advice and encouragement during this project.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The design of a general purpose graphic interactive modeling and simulation system is now being conducted by a group at the Department of Computer Science at the University of Illinois.[1] An example of the type of problem that the system will be able to handle is a network analysis. The graphic display will be used to specify a netowrk's elements and connections, and for presentation of output data. Another possible application is simulation of flight control. The user may manipulate controls at a display terminal. A central computer, by using this control information and the flight history, will calculate the new flight status that is to be displayed to the user.

The display terminal and a central computer are the basic components of the system. Often systems have displays which are tied directly to central registers of the parent computer. To display a point, its coordinates are first loaded into the central registers of the computer. A display command is then executed which results in a point being flashed on the screen. The problem with this scheme is that the processor is tied up in generating displays. The situation seems even worse when considering the refreshing of a static display. It is a repetative operation that will occupy an entire processor full time.

The cost of the display terminal includes the time devoted to it by the general purpose processor. The configuration described above is an expensive one. The cost per terminal is not reduced even when several display units are attached directly to the same general purpose

processor. The task of maintaining the picture at each terminal consumes most of the processor time.

The problem is therefore to try to reduce the cost per terminal but at the same time to maintain the display capability. The solution presented in this paper is the introduction of an intermediate system that will generate and maintain the display at each terminal. The central processor has to generate the picture informtion only once and pass it to the intermediate system. The system includes a Display Processing Unit (DPU) and a memory. The DPU interprets the picture information and generates sequences of efficient display commands which are stored in the memory. This information is used to refresh a displayed image.

Consequently, the central processor is relieved from the task of picture regeneration and may dedicate most of its time to general purpose computations. By keeping the cost of the DPU as low as possible and by sharing the memory among several display terminals, the cost per terminal is reduced by a considerable amount. The central processor we are using is the PDP-8. In some respects, the Display Processing Unit (DPU) is an emulation of the DEC 338. The main difference between the two is the DPU's capability of driving several display terminals at one time. A head-per-track disk is used as the refresh memory. The paper emphasizes functions and logic circuits (like picture generation and data communication) which are

peculiar to the DPU. No details are given on hardware (like instruction decoding) which is similar to that of the DEC 338.[2, 3]

II. SYSTEM OVERVIEW

The display system configuration is described in Figure 1.

The Display Processing Unit (DPU) is the interface between the PDP-8 memory and the rest of the display system. Because this interface executes programs from memory, because it has an instruction repertoire, and because programs can be written for the display processor, it can be thought of as a computer. It is a peculiar kind of computer whose capabilities are oriented toward making pictures rather than performing computational tasks. Many of its instructions are descriptive of drawing operations rather than computational operations. Because the DPU operates in conjunction with a general purpose computer, "general" computing capabilities were not given to it. The DPU retrieves data from the PDP-8 core memory, decodes it and generates picture information. This information is then transferred via a core memory buffer to a track of the central disk buffer, thus releasing the DPU for other tasks. The DPU performs the same functions that the DEC 338 does, with the additional capabilities of controlling and refreshing the picture at as many as 16 display terminals. Programming compatibility with DEC 338 is maintained except for two additional IOT's. All commands are transferred to the DPU via the PDP-8 single cycle data-break system. The programmer may specify which terminal he wishes to access at any time.

A special hardware scheme is used to generate the incremental commands from the combinations of codes which form lines and characters.

The order of execution of all the Control State instructions, PDP-8 display oriented instructions and the picture generation, is determined by a central control unit. The unit is composed of several sequential control points which govern the data flow in the DPU.

An interrupt queueing scheme controls the data communication between the display terminals and the DPU.

A central disk buffer holds the picture information for up to 16 display terminals, with a single track of the disk being dedicated to each of the terminals. Each of the tracks has its own read-head and line driver. The disk rotates at 30 revolutions per second, so the picture on each scope is refreshed at 30-cycle rate. This rate is high enough to give a flicker-free display. The display terminals can be remotely located from the central disk buffer. At the data rate used, 10^6 points/second are plotted by the display terminal.

The picture on each track of the disk is specified in terms of simple 3-bit incremental command. These commands are decoded by a display control at each of the scope terminals where X-Y deflection signals are produced to position and display points. The commands allow the beam to be moved a certain number of units in any direction on a 1024 x 1024 grid (left-right, up-down, or diagonally) covering a 10" x 10" area on the scope face.

Information corresponding to the initial values of X and Y, four spot intensities, scale factor, P. B. enable, and joystick interrupt-enable are also stored on the disk. Zoom option information is transferred directly to the display terminal, so that frames may be expanded by a factor of two, four, or eight.

The operator of the CRT display can communicate immediately with the DPU and the PDP-8 processor by using the incremental joystick or the push-button control box.

The DEC 338, in order to generate typical pictures, uses about 30,000 moves per frame, and the picture has to be reconstructed and regenerated 30 times per second to avoid excessive flicker. Also, the 338 may access the PDP-8 via the data-break system only once in three PDP-8 memory cycles, degrading the speed of generating points and vectors which results in lower picture capacity. The DPU disk buffer contains about 100,000 bits or 33,000 instructions in a full track. A track of data corresponds to a picture containing 660" of drawings. This means 66 screen widths of horizontal or vertical lines (in scale of 2) or about 1500 characters, or a proportional mix.

Consequently, the DPU disk configuration with its independent picture refreshing capability, easily handles display files like those executed by the DEC 338.

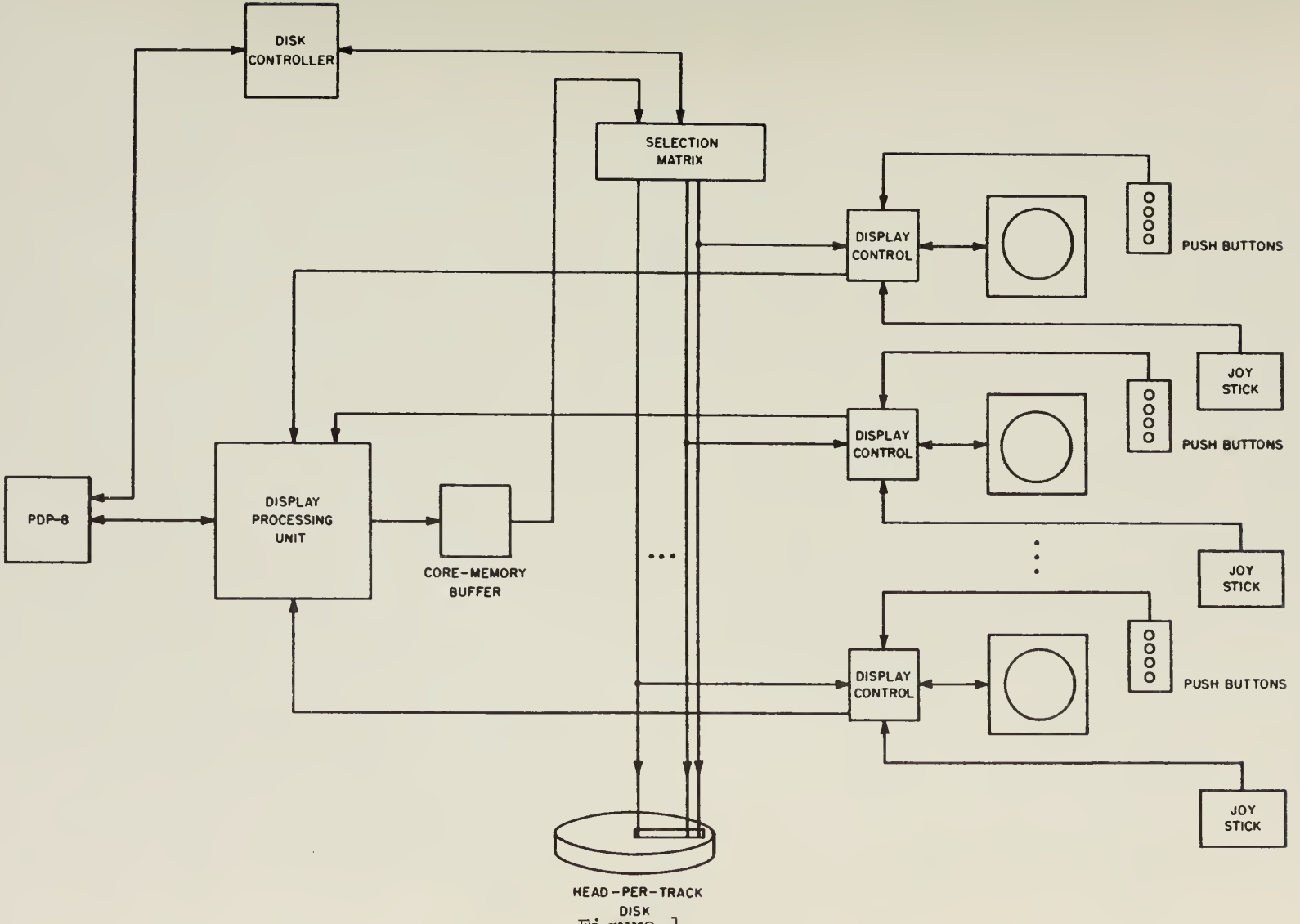


Figure 1

Display System Configuration

III. DPU REGISTER CONFIGURATION

The DPU has 18 internal registers which are of interest to the programmer. (See Figure 2).

The DPU also has an arithmetic unit (AU) and registers A, B, and D that are used during the process of generating the incremental commands. The interface of the DPU with its core-memory includes two 3-bit registers, a write buffer register (WBR) and a memory buffer register (MBR). The WBR is used to hold informtion that is to be sent to the external units through the DPU core memory and the disk.

Outlined below is a summary of the internal registers of the Display Processing Unit that can be accessed by the programmer, together with the particular function of each register.

DISPLAY ADDRESS COUNTER (DAC): The DAC is an up-counter that indicates where the next instruction is to be found in PDP-8 memory and is incremented immediately after an instruction is fetched from memory. It can be set by the INIT instruction, or it may be loaded immediately to affect a JUMP instruction, or its previous value can be pushed to affect a PUSHJUMP instruction. Associated with the DAC is a 3-bit break-field countup register, which allows the DPU to address any of 32K words of core-memory directly.

STACK POINTER (PTR): The PTR is a 12-bit up-down counter that points to the last filled location in a marked stack. In order to push an instruction into the stack, the PTR is incremented and the instruction is written at this location. The stack pointer operates in a manner opposite to that of the DAC, being decremented after it is used to fetch

an instruction from memory. This stack is the basic mechanism for accomplishing subroutining, since PUSHJUMP and POP instructions load and retrieve status data with the aid of the PTR.

DX AND DY REGISTERS: The DX register, containing 12 bits, is the main input register of the DPU. On data break, a data word from the PDP-8 memory is transferred into the DX register. From there it is transferred to one or more of the other registers of the DPU. Sometimes, information in the DX register is transferred to the DY register for buffering.

X AND Y REGISTERS: These are two 14-bit up-down counters containing the coordinates of the beam position at the time the picture is generated. The low-order 10 bits represents the actual beam position while the most significant 4 bits are used for validity checking in the case of an overflow.

X_{int} AND Y_{int} REGISTERS: At the time of an interrupt at one of the display terminals, the coordinates of the joystick marker on the screen are transferred into these two 10-bit registers.

DEVICE ID: This register contains the identification of the display terminal that requested the attention of the PDP-8 and the DPU.

X_d AND Y_d REGISTERS: These are two 3-bit registers containing information about the virtual screen size (page) as specified by the programmer. A page may contain up to 64 (8 x 8) screen sizes called

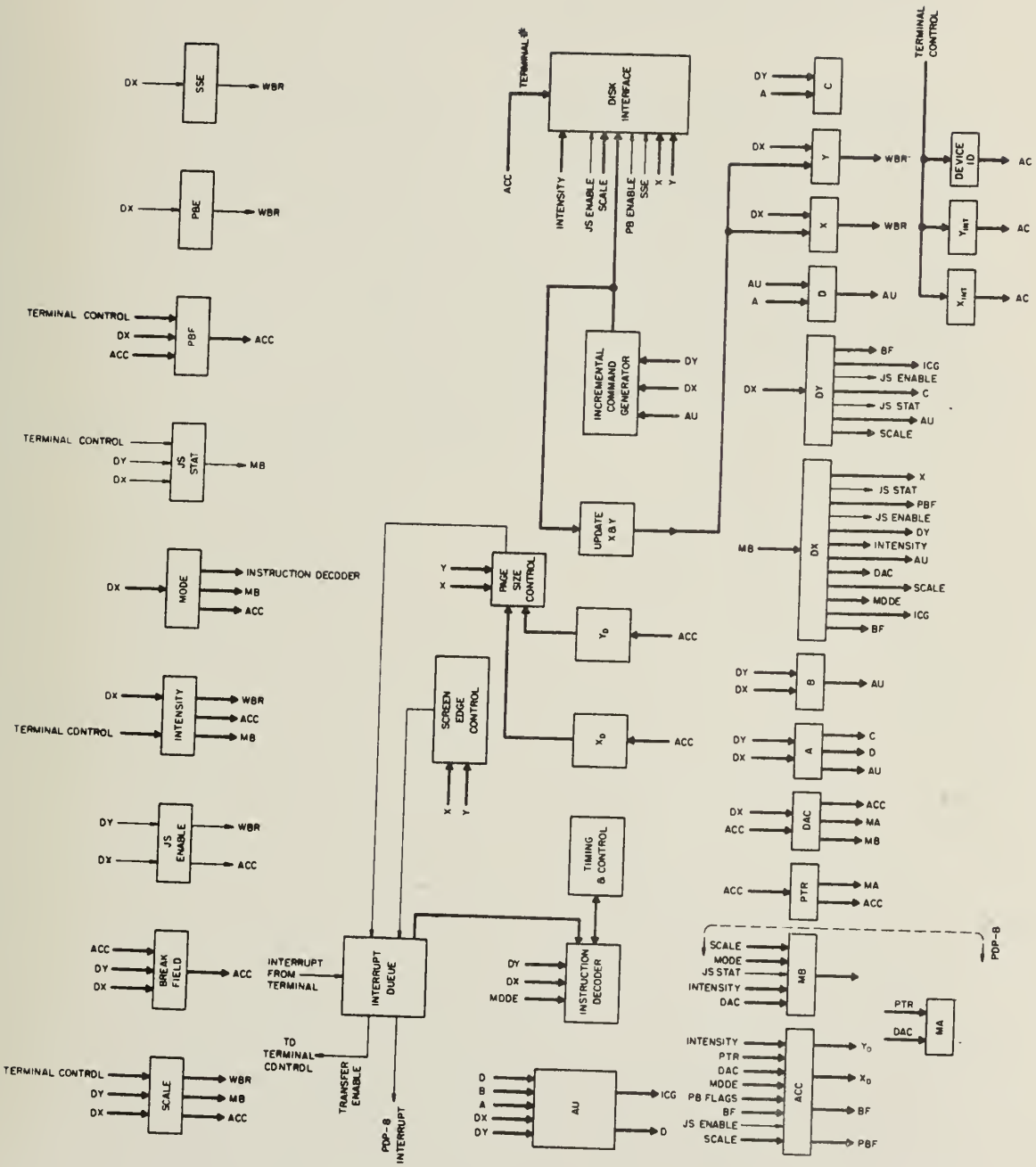


Figure 2

DPU Registers and Controls

sectors. This configuration allows a type of windowing in which the picture information in each sector may be displayed independently.

The other six registers contain information to set up the initial conditions of the DPU, such as enabling interrupts on special flags, setting the drawing size, the amount of intensification, etc. Some of this information will be stored on the appropriate disk track for use at the display terminal. The scale, intensity and PBF registers may also be loaded by the corresponding status information that comes directly from the display terminal.

IV. DISPLAY PROGRAMS

The display processor produces pictures by executing a program (i.e., a sequence of Display Oriented Computer instructions) fetched from a memory of the PDP-8. These PDP-8 instructions are executed as follows:

1. Control State instructions: In this state, instructions are executed in a manner similar to that of computer instructions. Each instruction has an OP code that determines the logic operations to be executed. The information transfer is done through DX and DY registers.

2. Data State instructions: The Mode register is set by Control State instruction before entering Data State and is used in order to interpret Data State instructions since they do not have an OP code. Their purpose is to control the motion of the CRT beam. Information transfer is done through DX and DY registers. A detailed description about the different modes of operation in this state and how they are executed will be given later.

3. Display Oriented Computer instructions: These are instructions that are used by the PDP-8 to communicate with the display. Any informtion transfer is done through the PDP-8 accumulator (ACC) where any required data must be before the IOT is given. In addition to the IOT's described in the DEC 338 manual, three others are peculiar to the DPU. The first is used at the head of display file to address the desired display terminal. The second is used when the PDP-8 is interrupted to identify the display terminal that initiated the interrupt request. Upon execution of this IOT, the information is transferred from the DEVICE ID register into the PDP-8 accumulator.

V. DISPLAY COMMANDS

The DPU interprets the picture represented by Control State and Data State instructions and translates this description into simple sequence of commands which are passed to the disk. The DEC 338 has seven data state modes, of which the DPU handles six: Point mode (0), Increment mode (1), Vector mode (2), Vector Continue mode (3), Short Vector mode (4), and Character mode (5).

Once the DPU enters Data State, information is obtained from registers DX or DY (or both) and is decoded according to the mode specified before entering this state. An escape bit "1" in one of the data words causes the display to return to Control State. Control State instructions are then executed until the system re-enters Data State by executing Mode or Pop instructions.

Each command passed to the disk is a 16-bit word which may be decoded in one of the following ways:

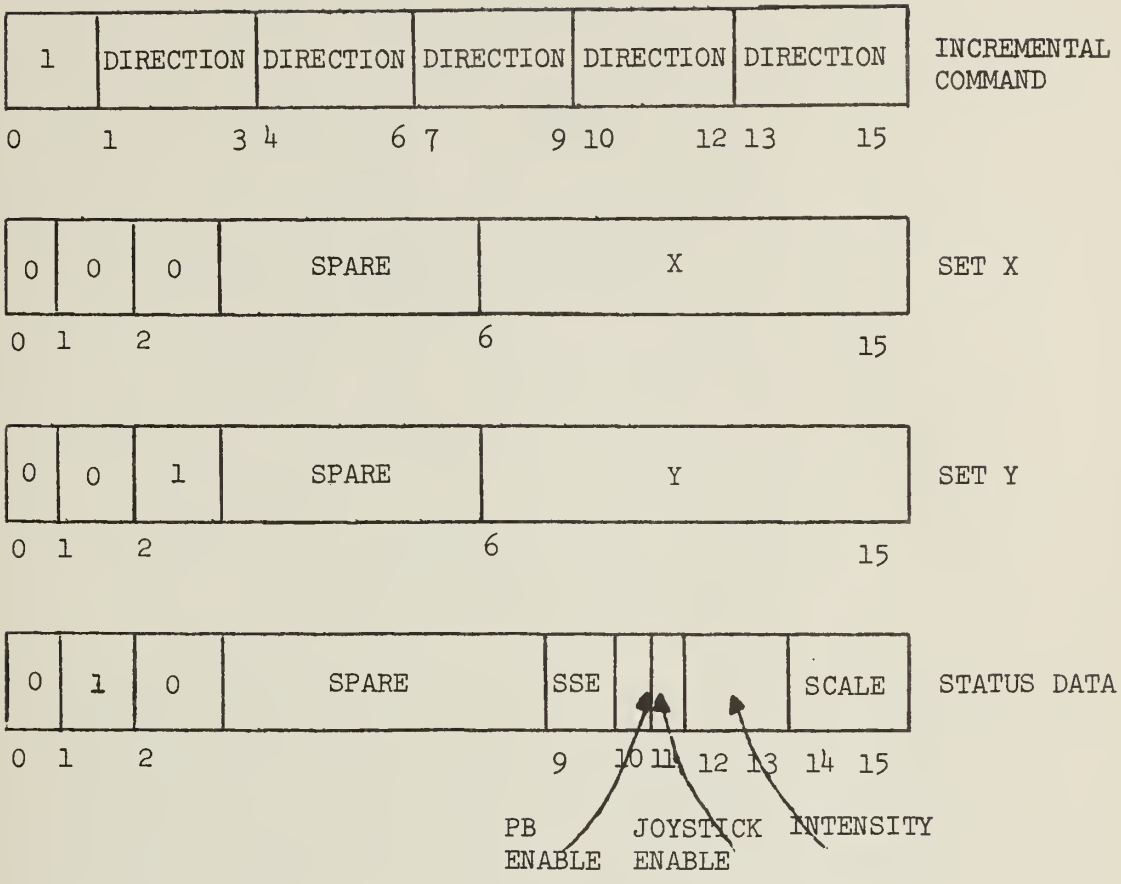


Figure 3

Display Commands

If bit 0 = 1, the word contains five groups of incremental commands. Each command is three bits indicating one of eight directions:

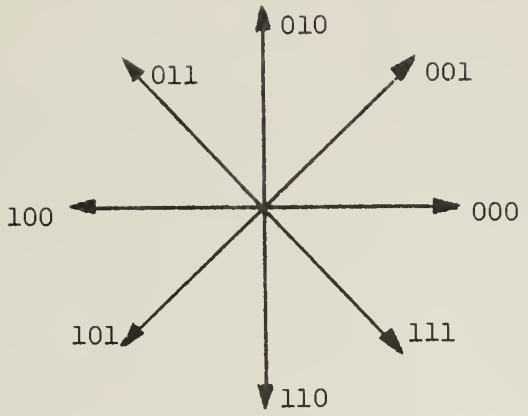


Figure 4

Incremental Commands

If bit 0 = 0, bits 1 and 2 specify a set X, set Y, or Status Data command. 00 in bits 1-2 means that the X coordinate of the beam is stored in bits 6-15. 01 in bits 1-2 implies that the Y coordinate is stored in bits 6-15.

The status data word is transferred to the disk track upon entering Data State. Bits 9, 10, and 11 will set the mask register which enables or inhibits Push-Button and Joystick interrupts at the display terminal. Bits 12 and 13 represent four spot intensities. Bits 14 and 15 are the scale factor (1, 2, 4, or 8). At the terminal end, all this informtion is read from the disk, and is decoded in order to control the picture on the screen. A detailed description of how each one of the first three words is generated follows.

VI. MODES OF OPERATION

A. Point Mode

This operation requires a pair of 12-bit instruction words for each point presented on the screen. These two words are transferred from the PDP-8 to registers DX and DY in two cycles. The execution during each cycle is described by the following flowchart in Figure 5.

B. Vector Mode--Incremental Command Generator

Vector, vector continue and short vector modes, and partly also increment mode use the Incremental Command Generator (ICG) logic. In order to approximate a vector the ICG produces a sequence of 3-bit commands. At the terminal end, each command will result in moving the beam in one of the eight directions (Figure 4). The beam is intensified at the end of each move. The sequence of moves represents a vector which is a close approximation of the theoretical (straight) vector. The algorithm employed by the ICG assures that the approximation will deviate from the straight vector by an amount not larger than one half unit of the CRT grid (in the scale of 1). The end points of the vector are always correct.

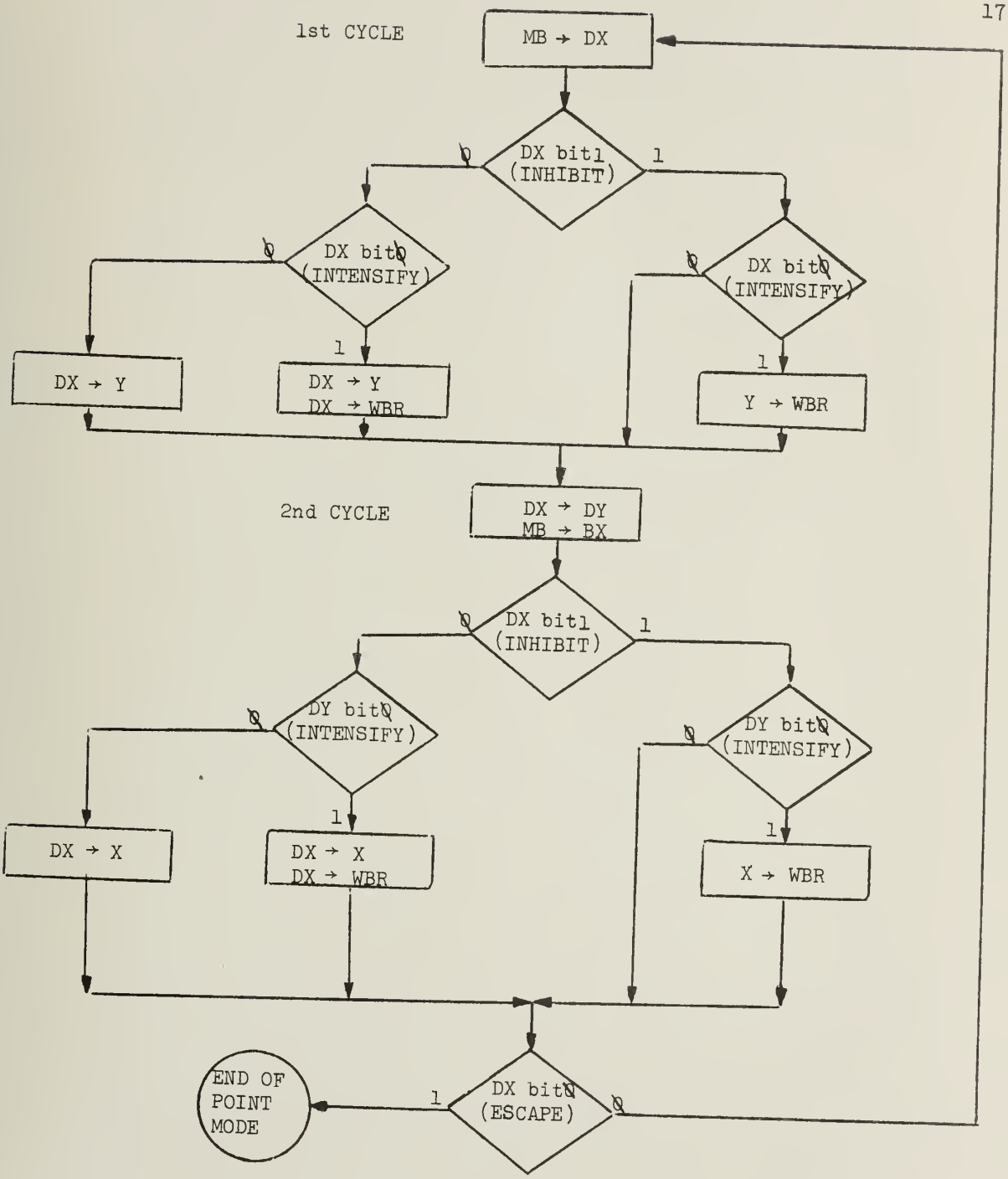
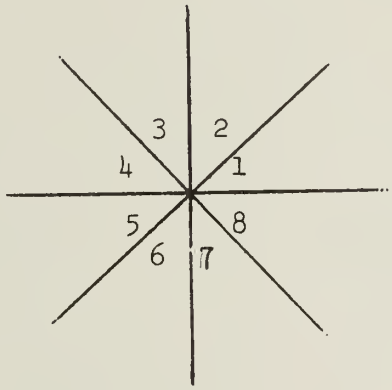


Figure 5

Point Mode Operations

The algorithm is described by the flowchart in Figure 6a. First, the region of the coordinate system in which the theoretical vector lies, is defined by the slope of the vector. More explicitly, we would like to know whether the vector is either in octants 1, 8, 4, 5, or in octants 2, 3, 6, 7.



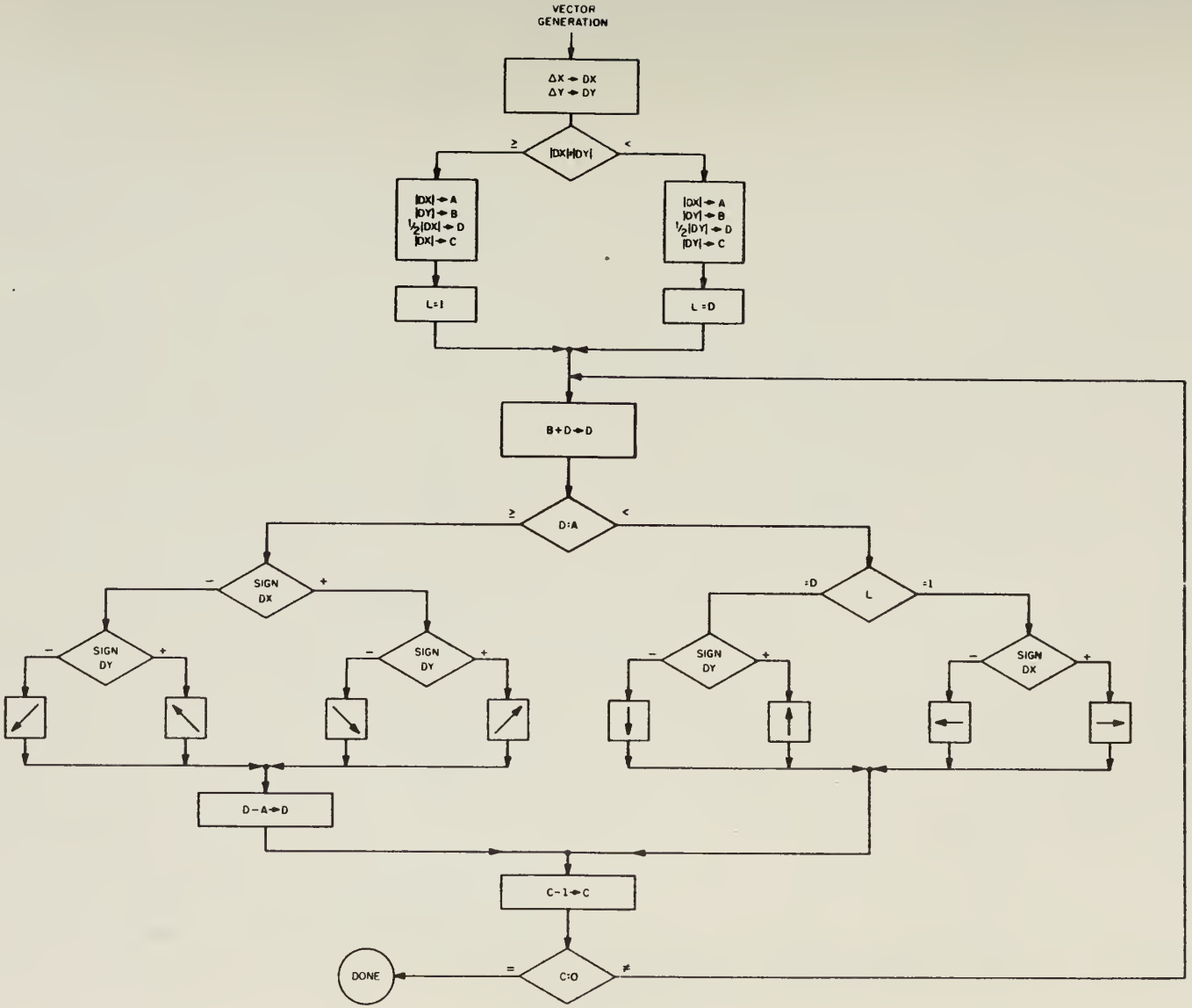


Figure 6a

Incremental Commands Generation-Algorithm

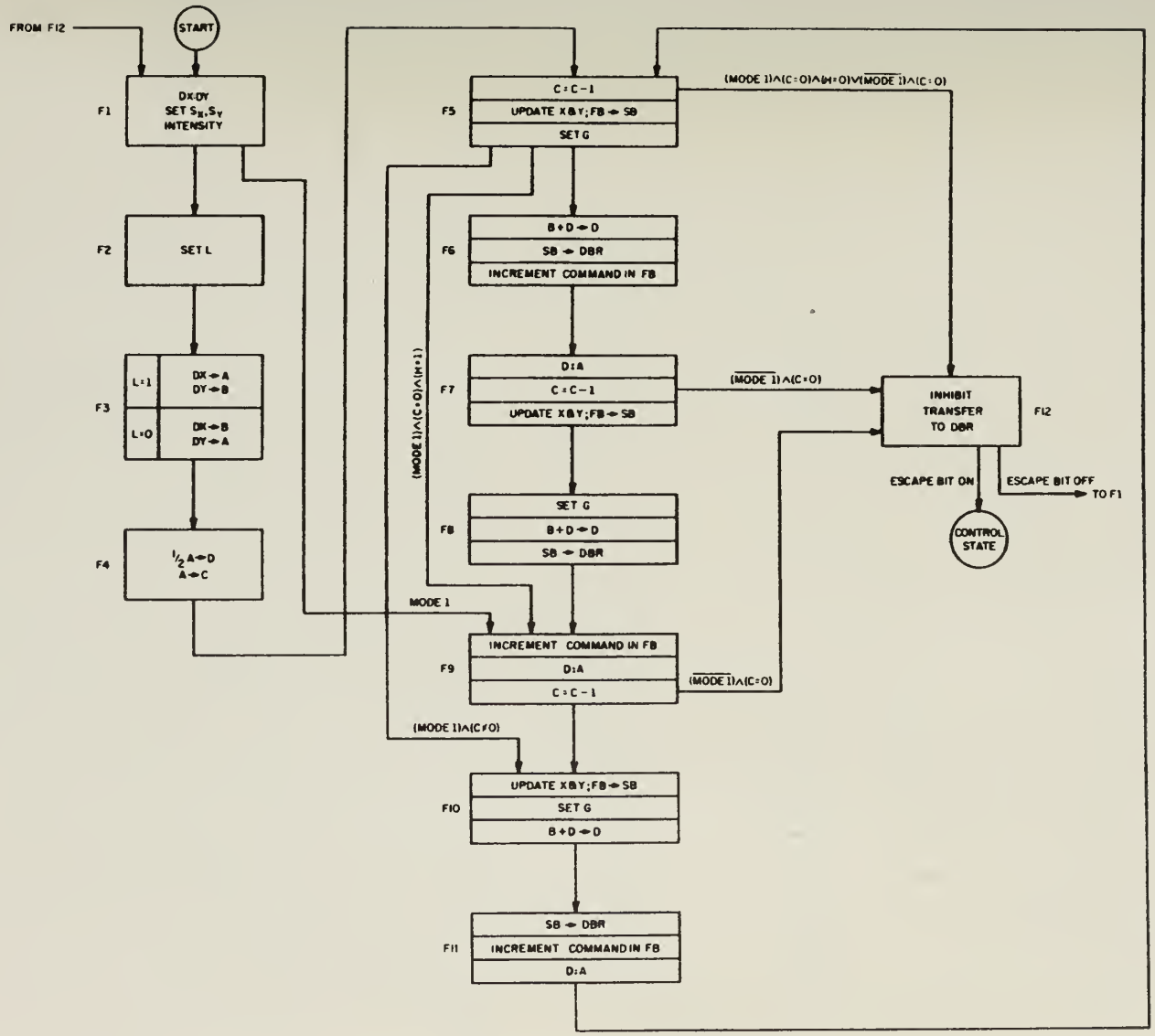


Figure 6b

Incremental Commands Generation-State Diagram

The computational part of the process is the same in both regions. This is accomplished by interchanging the role of the changes in the X and Y directions (ΔX and ΔY). The number of commands (or moves) that will approximate the vector must comply with the number of units of the largest between ΔX and ΔY.

For simplicity, let us follow the computation for a vector which lies in the first octant (its slope is less than or equal to 45° -- X = 9, Y = 2). Here nine steps are required and the move in each step is either → or ↗.

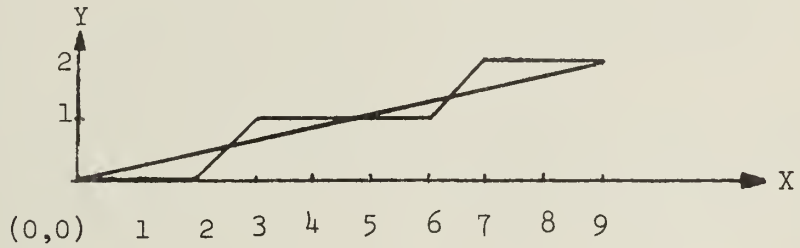


Figure 7

Approximation of a Vector Between (0,0) and (9,2) by Using the ICG Algorithm

As stated before, the approximation will not deviate from the straight vector between (0,0) and (9,2) by more than half a unit. This is accomplished by checking at every interval of X whether the Y coordinate of the theoretical vector exceeds a certain threshold level. If it does, the move is diagonal. Otherwise, it is horizontal. The initial threshold level is $\frac{1}{2}$, and it is increased by one every time Y is incremented (diagonal move). In this fashion we perform in each interval move, whose end points are the closest to the original vector.

At first, we would like to approximate the portion of the vector while $0 \leq X \leq 1$. If $y(X = 1) > \frac{1}{2}$, a diagonal move is made.

Otherwise the move is horizontal. In this example $y(X = 1) < \frac{1}{2}$ so that a horizontal move is performed. In the second step, the vector in the interval $1 \leq X \leq 2$ is approximated by comparing $y(X = 2)$ with $\frac{1}{2}$ resulting once again in a horizontal move. In the next step, the interval of interest is $2 \leq X \leq 3$. $y(X = 3)$ is greater than $\frac{1}{2}$ so that a diagonal move takes place and the threshold level is increased to $\frac{3}{2}$. The process continues in this manner. The resulting approximation is described in Figure 7. The ICG logic is divided into two parts--the first which generates the incremental command and the second which updates registers X and Y and performs additional control functions. (See Figures 8-11).

The hardware for the first part includes an adder, a counter (C), three 10-bit registers (A, B, and D), two control flip-flops (L and G), decoding logic and a 3-bit buffer register.

During vector mode, registers DX and DY are loaded with the values of ΔX and ΔY. The changes ΔX and ΔY, representing a vector, will be converted into a series of moves that will approximate this vector. Also loaded are the signs and intensifying bits of the changes. After the two load cycles, the ICG enters state F1. (See Figure 6b). The contents of DX is moved to M1, and that of DY to M2. Subtraction is used to compare the absolute values of the changes in each direction. The existence or absence of an overflow will set or reset flip-flop L. This occurs in state F2. In state F3, the contents of DX ($|\Delta X|$) is transferred to register A or B depending on whether L is "1" or "0". The contents of DY ($|\Delta Y|$) is moved to B or A following the same condition. Upon entering state F4, register A will contain the largest value between ΔX and ΔY. This value is also stored in counter C which will control the number of

moves required in order to approximate the vector. Half of the contents of A is stored in register D which will determine later in the process whether a move has to be in a diagonal or horizontal (vertical) direction. In state F5 the counter is checked for zero, and if it is not, the ICG will proceed to F6. Here the contents of B is added to that of D, and in state F7 it is compared with A. This is done in order to find whether the slope of the original vector is smaller or greater than $\frac{1}{2}$. This fact (G = 0,1) together with the signs of ΔX and ΔY (S_x and S_y) and the status of L are the inputs to the decoding logic which in state F9 generates the 3-bit incremental command. This command is stored temporarily in a register.

The hardware of the second part includes counters X and Y (14 bits each), scale logic, and two 3-bit registers X_d and Y_d. The two counters simulate the coordinates of the beam as if a real picture is being displayed at the same time the picture information is generated. They are updated as a result of the incremental command in the following manner: During state F9, when the code is generated, it is applied to a combinational logic circuit. This circuit decides whether X and Y should be incremented or decremented. This information is fed to scale logic where the scale factor desires (1, 2, 4, or 8) has been stored before. The result is that the increment or decrement pulse is applied at state F10 to one of the four least significant positions of the counter. The updated contents of the counters is checked now to see whether the beam position they are simulating is outside the specified page limits. If it is, an edge interrupt is then issued.

The transfer of the incremental command to a second buffer is inhibited if one or more of the following conditions holds:

1. Edge interrupt (HEF or VEF are ON).

2. An attempt is made to position the beam outside the screen limits.

3. The beam in its new position inside the screen should be OFF.

In cases 2 and 3, the PM flip-flop is set to inhibit the transfer. Consequently, the nonintensified moves, or moves beyond the screen limit will not be recorded on the disk. If the conditions mentioned above do not exist, the PM flip-flop is checked to see if it is in "1" state. If it is, this means that the previous move was not recorded on the disk. In this case the beam has to be brought to a position where the display will start. This is done by transferring the updated values of X and Y counters to the disk rather than the incremental command of the last move. Otherwise the transfer of the incremental command is enabled by a pulse TIC. At F11 the three bits are moved to the Direction Buffer Register (DBR) and the ICG goes back to state F5.

In order to gain speed when translating the vector representation into a series of incremental commands, many of the operations starting at state F5 may overlap. When the first command has to be generated, the ICG enters state F5 and performs the operations as described above. States F6 and F7 will follow if the contents of counter C is not zero. At that time, an additional attempt to generate another move is initiated by checking the contents of C and proceeding in the same way as the first time. A third attempt is made in state F9. From there on the ICG will perform three different operations in parallel. While in state F10, for example, the updating of the X,Y

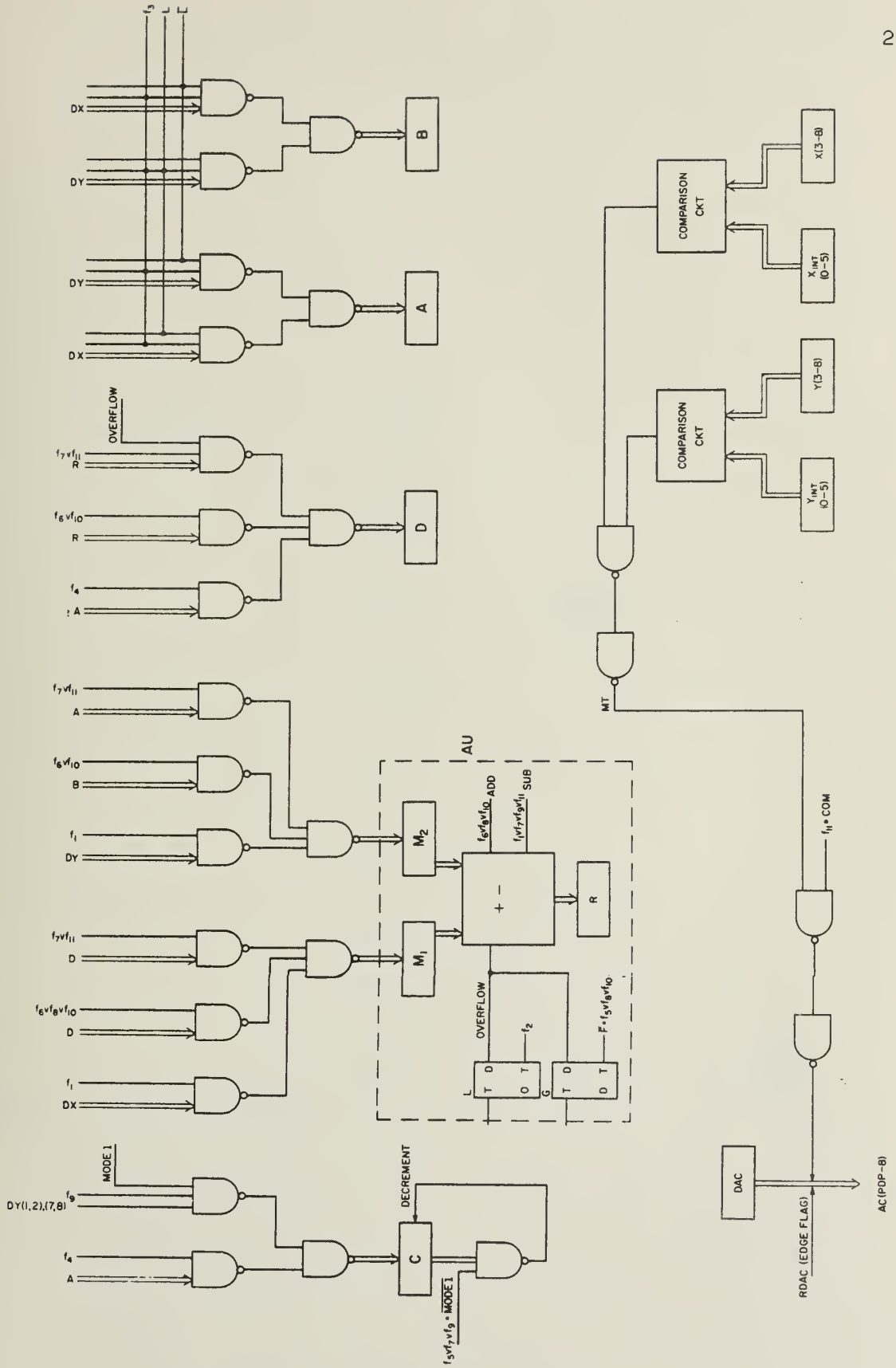


Figure 8

Incremental Command Generator Arithmetic (Schematic)

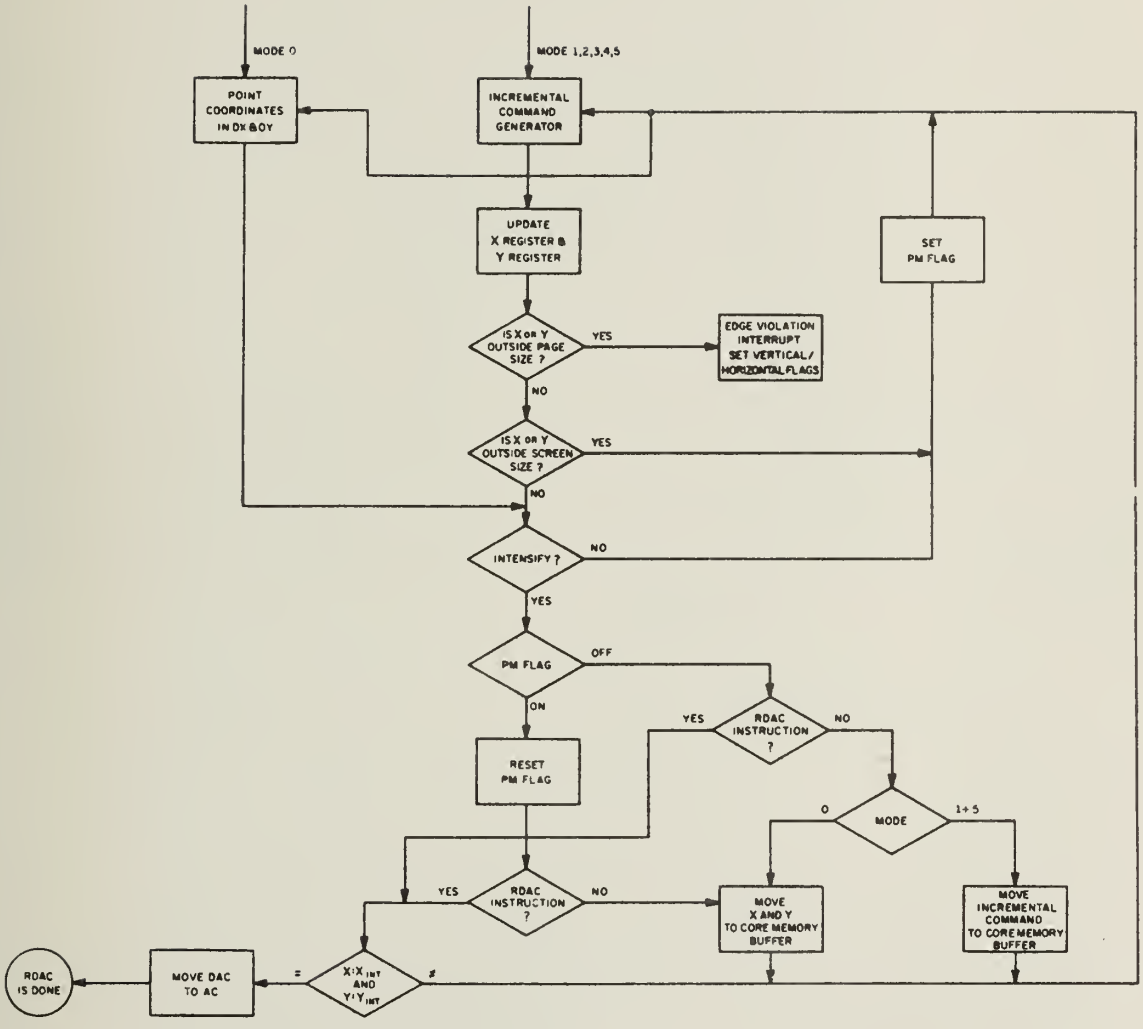


Figure 9

ICG-Page and Screen Size Control (Flowchart)

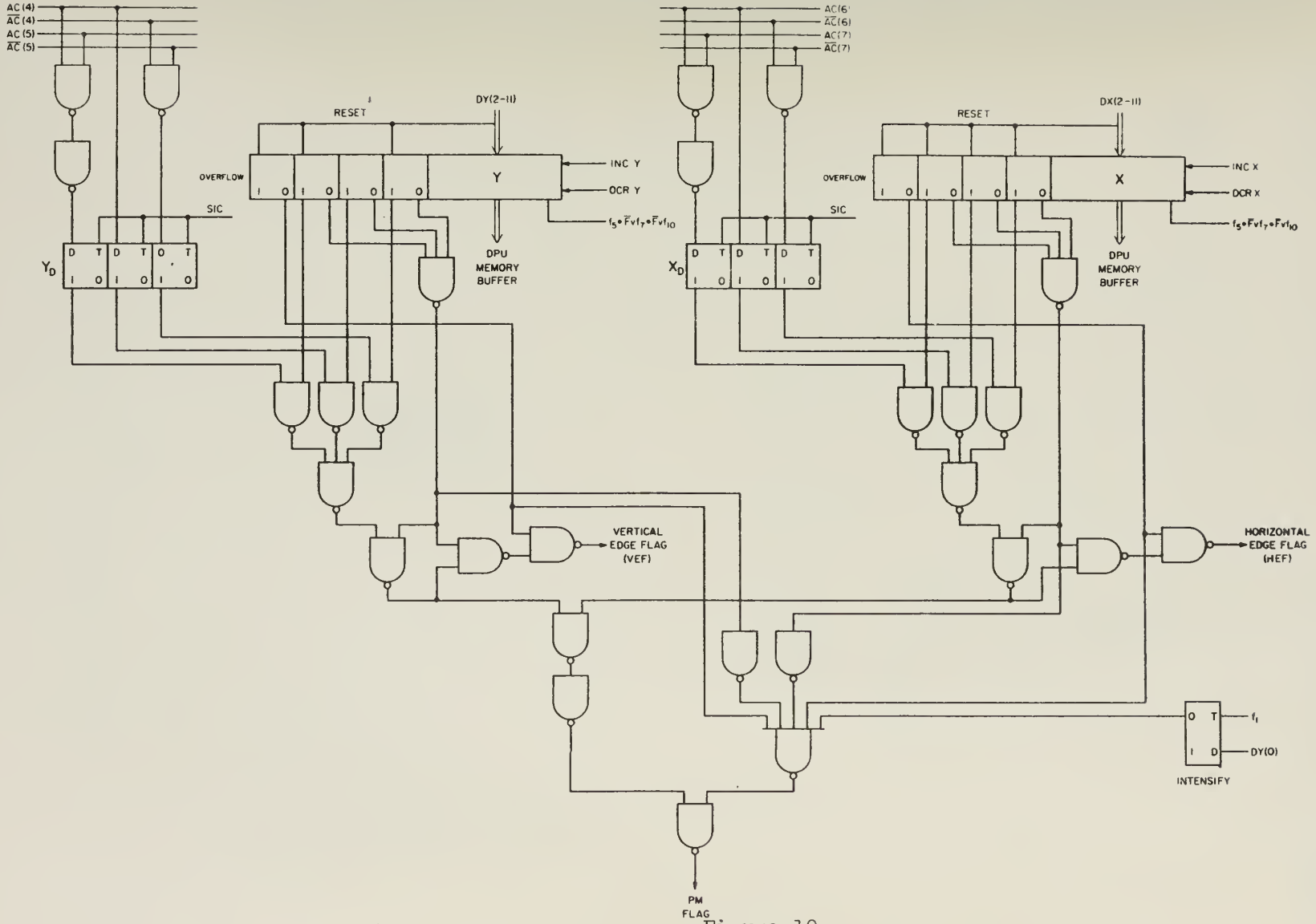


Figure 10

ICG-Page and Screen Size Control (Schematic)

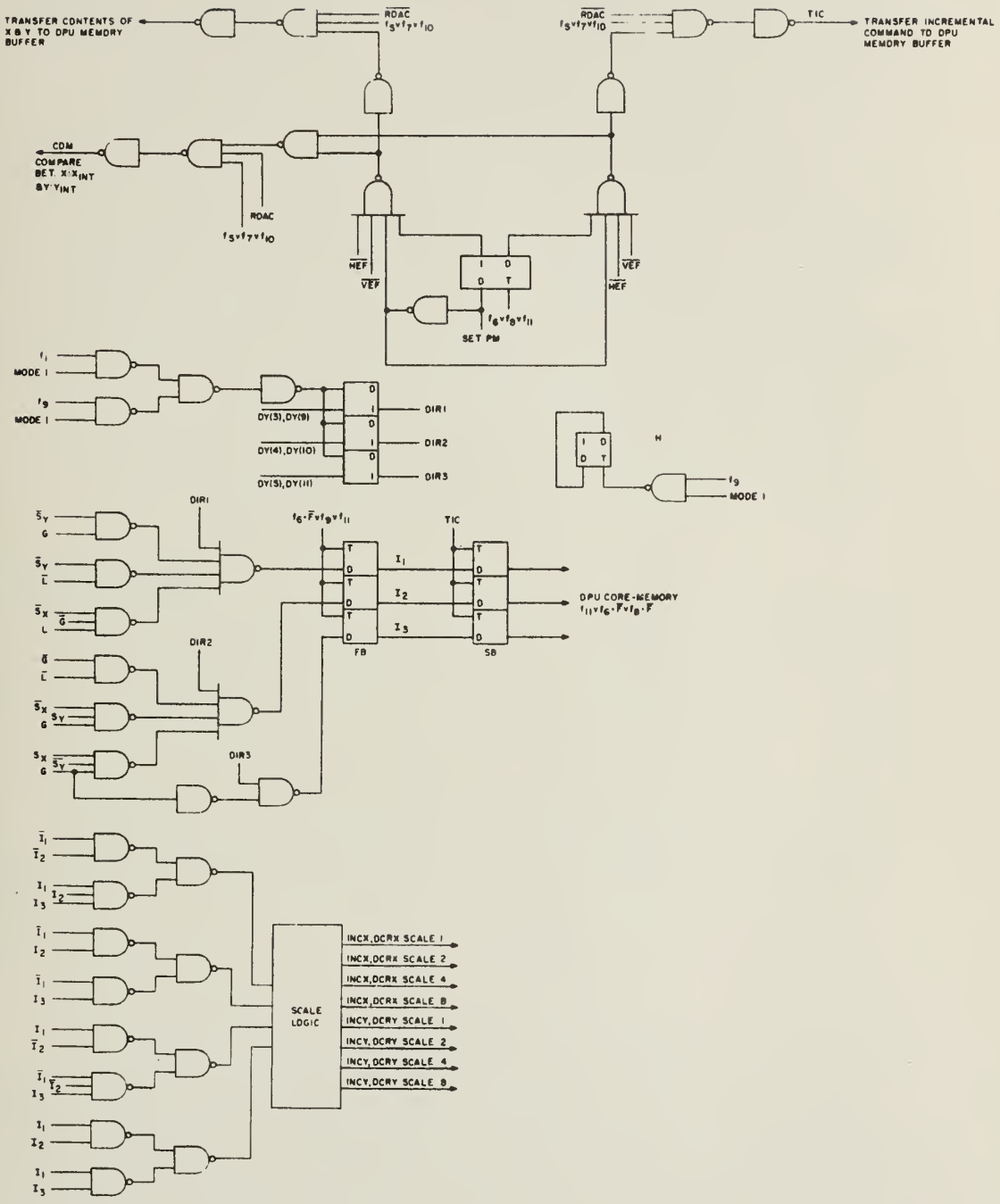


Figure 11

ICG-Code Generation and Transfer Commands

counters and the checking for irregular conditions is done for the first command now temporarily in the first buffer. At the same time, flip-flop G is set or reset in conjunction with the second command, and the operation B + D → D is performed for the third one. Once counter C is found to be zero, say while the ICG is in F7, the operations will continue until the ICG reaches state F5. The normal sequence will be terminated, and the ICG goes to state F12. Any transfer to the DPU core buffer is inhibited. If the escape bit is off, the ICG will return to state F1; otherwise, the next PDP-8 instruction will not be implemented as a data state instruction.

C. Increment Mode

Instructions in this mode are executed by part of the ICG hardware. The sequence of operations is as follows:

State F1: The contents of register DY, bits 3-5, is transferred to a 3-bit register, and bits 1-2 (number of moves) to counter C.

State F9: The 3-bit direction code is gated to a second register. This is the same one used in vector mode to store the incremental command. A flip-flop H is set to indicate that the first half of the instruction is being executed. The code representing the direction of the move, as stated by the programmer, will be treated in the same manner as the incremental command generated by the ICG.

States F10 and F11: The ICG will function the same way as described above. The ICG proceeds to state F5 and the counter C is decremented. If it is greater than zero, the next state is F10; otherwise, the next state is F9. Flip-flop H is reset, and information of the second half of the instruction is gated as in state F1. The execution of the second part of the instruction is terminated when the counter C, which is checked in state F5, is zero. The ICG then proceeds to state F12.

VII. INTERRUPTS

One or more display terminals may require the attention of the PDP-8 and consequently that of the DPU. There are several conditions, most of them related to the position of the beam on the screen or to operator initiative at a particular display terminal, that will set display flags that can interrupt the PDP-8. The states of display flags are sensed by IOT skip instructions.

The flags used have the same interpretation as in DEC 338:

1. Edge violation.
2. External stop.
3. Joystick (light-pen).
4. Push button hit.
5. Manual interrupt.

Unlike the DEC 338, it is not necessary to stop the display after an interrupt.

Flags 1-4 may be cleared by one of the following PDP-8 instructions: RES1, RES2, and CFD. Unlike its use in usual PDP-8 programs, internal stop command must be at the logical end of the display program. It signals the DPU to terminate the display file for which it is now generating incremental commands. The DPU then interrupts the PDP-8 with this information.

Violation of the screen edge is detected while generating the incremental commands. The programmer is expected to determine the size of the page by specifying the value of bits 4, 5, 6, and 7 in SIC instruction. This information is decoded into two 3-bit registers X_d and Y_d:

000 - 9.375" (10 bits)

001 - 18.75" (11 bits)

011 - 37.5" (12 bits)

111 - 75.0" (13 bits)

The updated counters X and Y are checked every time to see that their values do not exceed the indicated page size. The operation is described by Figure 9 and the logic circuit for it is in Figures 10 and 11. As described above, one of three cases may exist:

1. The limits of the page were violated. A vertical or horizontal edge flag or both will be set, causing an interrupt.

2. The screen limits were violated but the coordinates as specified by X and Y registers are still within the page size. The incremental command which represents the current move will not be transferred to the disk. This process continues until the designated position of the beam is found to be once again inside the screen. Its coordinates, rather than the last increment code, will be stored on the disk for display. In this way, space on the disk track is saved by not storing nondisplayed picture information.

3. Neither of the previous conditions exist. The coordinates represent the beam position within the screen, and the incremental command is moved to the DPU core-memory and later to the disk. The manner in which other flags are detected will be described later in this section.

A. Interrupt Queueing

The PDP-8 may be interrupted by various I/O devices or conditions internal to the processor. The interrupt request is issued via the PDP-8 interrupt bus. The processor may then determine the cause of the interrupt by sensing the flags, including those mentioned before.

In the multi-terminal configuration we are facing here, the PDP-8 first has to realize that the interrupt call comes from the DPU or one of its affiliated display terminals. Secondly, it must identify the terminal causing the interrupt, and then sense the various flags associated with that display terminal.

More than one display terminal may try to interrupt the PDP-8 at one time. It is the responsibility of the DPU to queue those requests and enable the PDP-8 to respond to them one at a time. A special interrupt queueing scheme is devoted to this task. The logic circuit is shown in Figure 12.

Each display terminal contains an Interrupt Flip-Flop (IFF). When a particular condition has occurred (joystick hit, push-button, manual interrupt), the IFF is set. There is also a mask register (not shown in the figure) which can inhibit certain classes of interrupts. The mask register, which may be loaded by a SIC instruction or altered by commands on the disk, allows the user to determine which terminals are to respond to particular conditions. The setting of any terminal IFF will set the Central IFF (CIFF). If the PDP-8 is not busy servicing another interrupt request, all the IFF's will be scanned sequentially by a logic circuit.

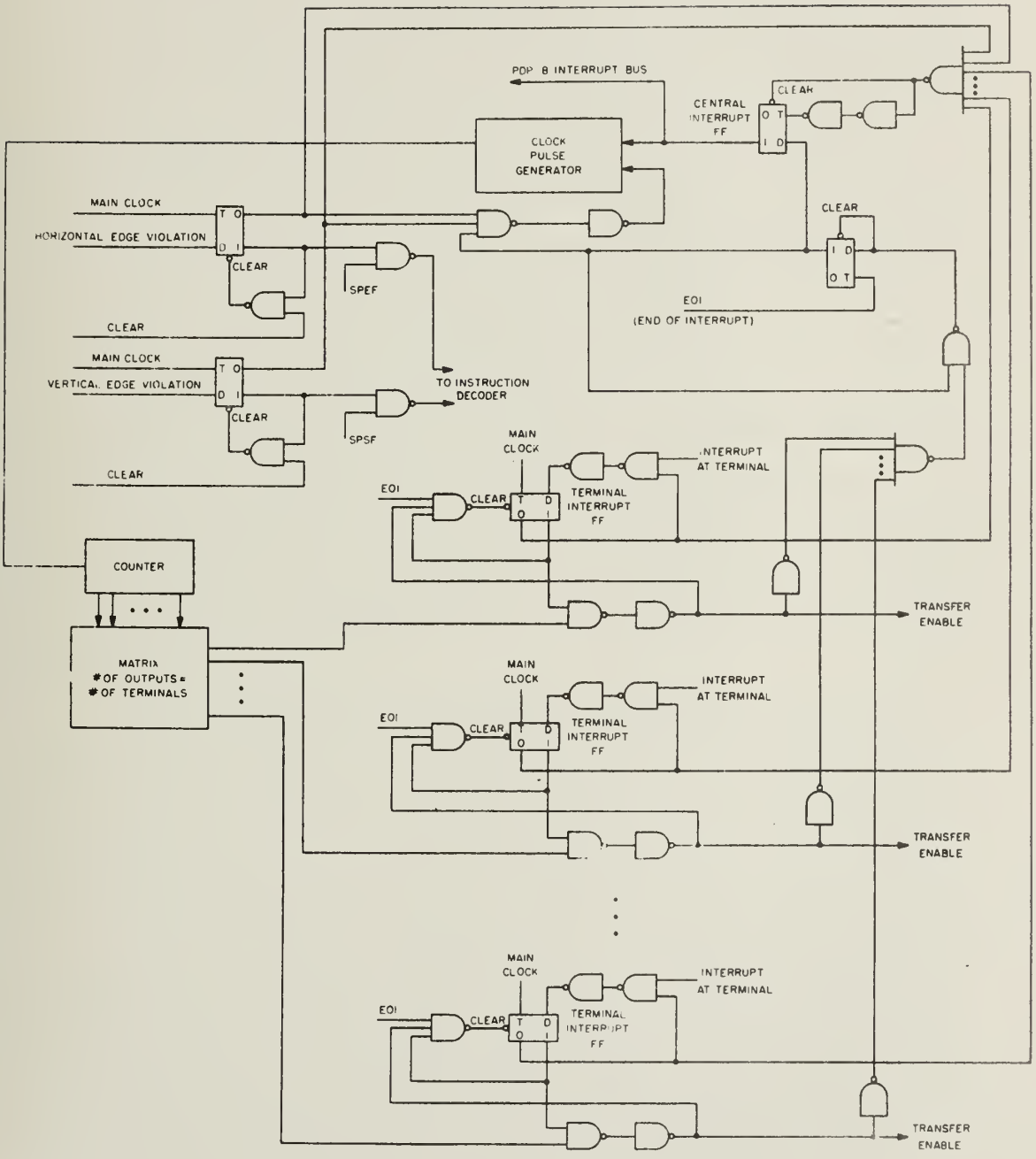


Figure 12

Interrupt Queue

This scanner is composed of an oscillator, a counter, and a matrix. The oscillator will start to generate clock pulses when both the following conditions are satisfied:

1. One or more terminals have issued an interrupt request.

2. The PDP-8 has completed executing a previous request.

The clock pulses increment the counter whose output is applied to the matrix. The number of output lines from the matrix is identical to the number of terminals. Each line is gated together with the output from an IFF of the corresponding terminal.

Once the matrix detects a terminal IFF in the "1" state, the oscillator and consequently the counter are stopped. Hence, the contents of the counter specifies the interrupted terminal.

The PDP-8 will later issue a special IOT (does not exist in DEC 338 repertoire) and the contents of the counter will be passed as a Device ID.

The signal "Transfer Enable" will transfer the status registers of the appropriate terminal to the corresponding registers in the DPU. The X and Y coordinates of the joystick marker at the time of interrupt will be stored in registers X_{int} and Y_{int}, respectively.

While this interrupt condition is being handled by the PDP-8, any other request from any terminal will be queued by setting the pertinent IFF and the CIFF. The PDP-8 terminates processing of the current interrupt by sending an EOI (End Of Interrupt, RES1 or RES2) signal. This will reset the Interrupt Flip-Flop and an attempt will be made to reset the Central FF if no other interrupt condition is pending. If there is one waiting, or if a request is issued later, the scanner will resume from the point where it stopped.

An edge violation interrupt has priority over interrupt requests coming from the display terminals. It inhibits the scan and is serviced as soon as the PDP-8 is ready. When the flag is cleared, the scanner is activated again.

B. Programming Compatibility Problems

As mentioned before, the DPU executes the display file that is fetched from the PDP-8 core-memory, generates incremental commands, stores these commands temporarily in the DPU core-memory, and upon termination of the display file transfers a block holding the commands to a track of the disk buffer. The display then starts by reading the commands off the disk track, and the DPU will start to generate another display file, maybe for another terminal. In the DEC 338 picture generating and displaying on the screen are carried out simultaneously.

This difference in concept between the DEC 338 and the DPU may lead to some difficulties. The contents of several registers in the DPU which the program may like to interrogate are lost. This problem becomes crucial at the time of an interrupt. If a RDAC instruction is issued, the contents of the DAC register has to be transferred to PDP-8 accumulator in order to identify the last command that was executed before the interrupt occurred. In addition, the program may like to know the mode, breakfield, etc. Yet, DAC, BF, and the Mode registers do not contain the proper information.

The problem was solved by executing the RDAC in the following fashion. (See Figures 9 and 11).

1. If the cause for interrupt is a vertical or horizontal edge violation, the contents of the DAC is the one we are looking for because the edge violation was detected while generating the incremental code. It is transferred to the AC in the PDP-8.

2. If the cause for interrupt is other than edge violation, a search is made to find the contents of the DAC by performing a "dummy generation" of the picture displayed at the interrupted terminal. The initial address of the display file is found in the AC register (the program, recognizing the display that caused the interrupt, provided it before issuing the RDAC). The DPU will issue a break request resulting in the transfer of commands from the PDP-8 core-memory. From now on, the procedure for generating incremental commands is the same as described in Section III. The X and Y counters are compared every time with X_{int} and Y_{int} respectively. The low order four bits of each register are disregarded, allowing a resolution of 0.15". If both comparisons agree, generation is terminated, and the current contents of the DAC is transferred to the AC.

To summarize, the following status flags and registers may be sensed only after a RDAC has been issued because only then will they contain the proper information:

1. Sector Zero Flag.
2. Control State Flag.
3. Break Field Register.
4. Byte Flag.
5. Mode Register.

Others will be available at any time either because of their characteristics, or because the information required can be obtained directly from the assigned display terminal:

1. Joystick Hit Flag.
2. Vertical Edge Flag.
3. Horizontal Edge Flag.
4. Push-Button Hit Flag.
5. Manual Interrupt.
6. Display Interrupt Flag.
7. Joystick Enable.
8. Intensity Register.
9. Scale.

VIII. DPU--MEMORY--DISK INTERFACE

The various display commands generated by the DPU are stored on a track of a disk memory. The disk rotates at 1800 RPM. Consequently, the rate in which data has to be presented to the disk track is $3 \cdot 10^6$ bits per second. The DPU is unable to comply with this data rate. The rate at which commands are generated is not constant and depends on the type of PDP-8 display instruction being executed by the DPU. An intermediate device, a buffer, is needed to accumulate the display commands as they are generated by the DPU.

The DPU uses an AMPEX RF-1 magnetic core-memory for this buffering. It is a rapid access, compact, coincident current memory, employing modular construction. The storage capacity is 4096 16-bit words with a cycle of 2 usec in the "Read-Modify-Write" mode of operation.

The processor communicates with the memory through a memory buffer register (MBR) and memory address register (MAR). The DPU stores the commands temporarily in the memory. At the time the display file is terminated, the block of instructions is moved from the memory to the corresponding disk track.

Data flow and register configuration are described in Figure 13. The Set X, Set Y, and Status Data words are transferred directly into the 16-bit write buffer register (WBR). The incremental commands generated by the ICG in groups of three bits are first filed into a 16-bit word and are then transferred into the WBR. A modulo-5 counter, controlling this process, is reset initially when the generation of the previous display file has been terminated. The first time the ICG is in state F11, it will transfer the first incremental command. The decoding logic matrix

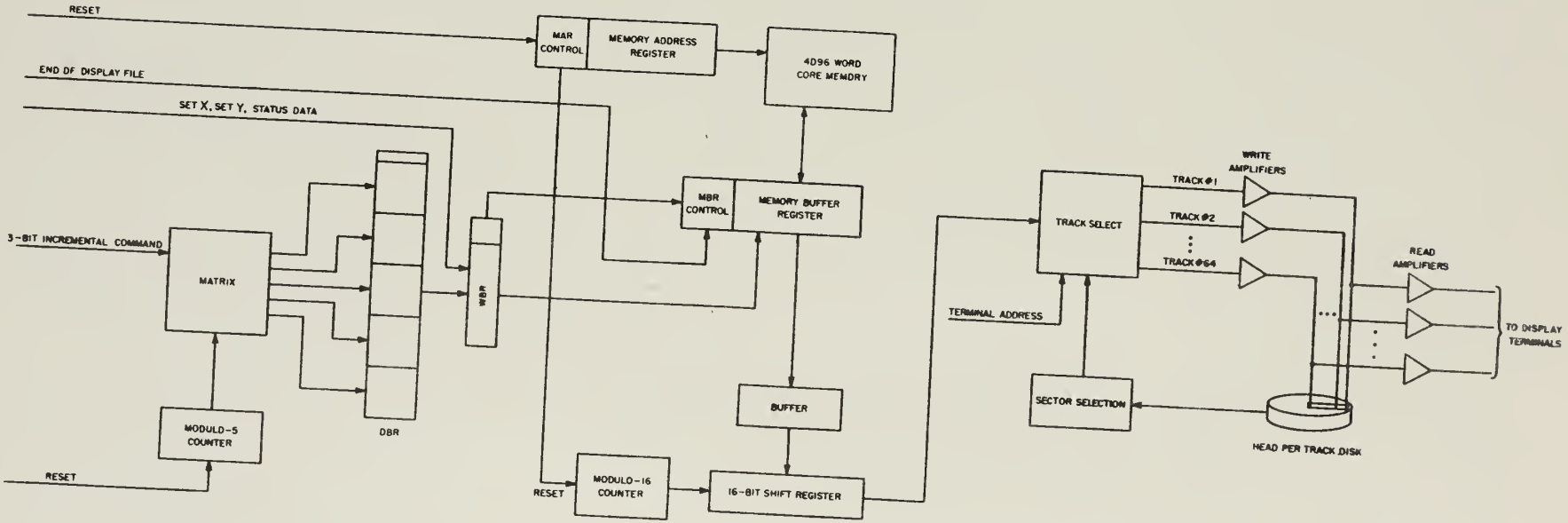


Figure 13

DPU-Core Memory-Disk Interface

which is controlled by the status of the counter will cause the command to be stored in bits 1-3 of the DBR. The counter is then incremented so that the next command is stored in bits 4-6. The process continues in this manner until the counter is reset. The contents of the DBR is moved into the WBR, and the MBR is signaled that information is ready to be stored in the core-memory. Bit 0 of the DBR is always 1.

When the ICG works at its maximum data rate, it will fill the DBR in 1 usec. This is too fast compared with the 2 usec memory cycle. In order not to over-run the WBR, one provision could be to inhibit the transfer of code from the ICG until the first half of the memory cycle has been performed. Another possibility is to buffer the incremental command at the output of the DBR.

The MAR is reset initially at the beginning of the generation of a display file. It is incremented as long as instructions are stored in memory, and is reset again by the IOT that signals the end of the display file. The same signal also initiates the transfer of code from the core-memory to the disk-track.

The DATA-DISC FPD disk was chosen as the rotating memory refresh mechanism. The disk contains 64 data tracks with 100,000 bits on each track. It rotates at 1800 RPM to provide a maximum bit rate of three million bits per second. The core memory communicates with the disk through a 16-bit buffer and a 16-bit shift register.

Upon completion of the transfer to the core-memory, the MAR is reset to point to location zero. At the same time the MBR is signaled to start reading data from the core-memory. The data is moved first to a buffer and then to the shift register. A modulo-16 counter controls

the shift to the track-select combinational logic, and is incremented by the clock track. Each track is devoted to one display terminal; the programmer specifies by an IOT the terminal to which this display file belongs. The track-select logic decodes this information and directs the 16-bit word entering serially into the appropriate track on the disk. This logic also enables the write amplifier of the corresponding track. The storing process will start when the write head is positioned at the beginning of the track. The 16-bit word will be stored on the disk in about 5 usec. After eight counts, the module 16 counter signals the MBR to read another word from core-memory, so that by the time the 16 bits are shifted, a new word is ready at the buffer to be moved to the shift register. The process continues in this fashion until the entire block of instructions is moved out of the memory and stored on the track.

IX. DISPLAY CONTROL

A block diagram which describes the display control is presented in Figure 14. The three main functions are:

1. To read the 16-bit word from the disk track, decoding it and updating X, Y, Scale, and Z (intensity) registers.

2. Displaying the joystick marker.

3. Controlling and executing interrupts caused by the joystick and push-buttons.

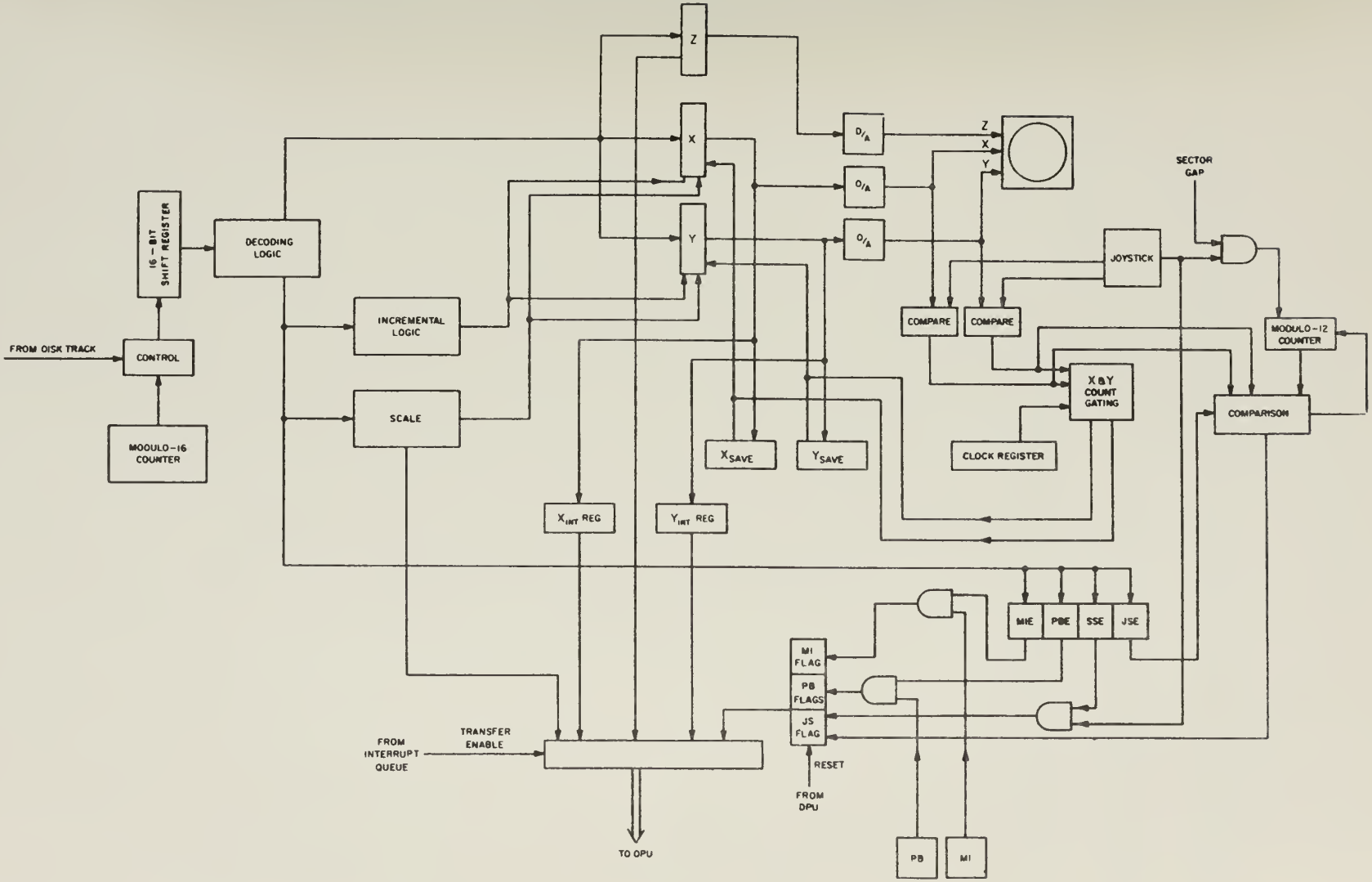


Figure 14

Display Control

LIST OF REFERENCES

1 Gear, C. W. An Interactive Graphic Modeling System, Department of Computer Science Report No. 318, April 1969, University of Illinois, Urbana, Illinois 61801.

2 Programmed Buffered Display 338 Programming Manual, DEC, Maynard, Massachusetts, 1967.

3 DEC 338 Buffered Display Instruction Manual, Maynard, Massachusetts, 1967.

APPENDIX

CONTROL STATE SUMMARY

Op Code: Parameter			Scale			Light Pen		Intensity			
0	1	2	3	4	5	6	7	8	9	10	11
0	0	0									

Op Code: Mode			Stop Code	Clear Push-Button Flag	Mode			Clear Sector Bits	Clear Coordinate Bits	Enter Data State
0	1	2	3	4	5	6	7	8	9	10	11
0	0	1									

First Word

Op Code: Jump			Scale			Light Pen		Push	Break Field		
0	1	2	3	4	5	6	7	8	9	10	11
0	1	0									

Second Word

Low Order			12 Bits of Address								
0	1	2	3	4	5	6	7	8	9	10	11

								Inhibit Restoring			
Op Code: Pop			Scale			Light Pen		Mode	Light Pen and Scale	Intensity	Enter Data State
0	1	2	3	4	5	6	7	8	9	10	11
0	1	1									

Op Code: Conditional Skip (Bank 1)			Sense of Test	Clear Bits After Test	Complement Bits After Test	Selected Buttons 0-5					
						PB0	PB1	PB2	PB3	PB4	PB5
0	1	2	3	4	5	6	7	8	9	10	11
1	0	0									

Op Code: Conditional Skip (Bank 2)			Sense of Test	Clear Bits After Test	Complements Bits After Test	Selected Buttons 6-11					
						PB6	PB7	PB8	PB9	PB10	PB11
0	1	2	3	4	5	6	7	8	9	10	11
1	0	1									

Op Code: Miscelloneous			Microprogrammed: Arithmetic Compare PB (0-5)			Push Buttons (0-5)					
						PB0	PB1	PB2	PB3	PB4	PB5
0	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	0	0						

Op Code: Miscelloneous			Microprogrammed: Skip on Flags			Skip Unconditional	Skip if not Sector 0	Skip on Push-Button Hit Flag		Spare	
								Bank 1	Bank 2		
0	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	1	0						

Op Code: Miscellaneous			Micro-pro: Sloves	Group Number		Unit 0			Unit 1		
0	1	2	3	4	5	6	7	8	9	10	11
1	1	0	1								

DATA STATE SUMMARY

First Word (DY)

Point (000)											
Intensity	Inhibit	Y Position									
0	1	2	3	4	5	6	7	8	9	10	11

Second Word (DX)

Point (000)											
Escape	Inhibit	X Position									
0	1	2	3	4	5	6	7	8	9	10	11

Increment 001											
Intensity	No. of Moves		Direction (0-7)			Intensity	No. of Moves		Direction (0-7)		
0	1	2	3	4	5	6	7	8	9	10	11

First Word (DY)

Vector (010)											
Intensify	+ −	10-Bit Delta Y									
0	1	2	3	4	5	6	7	8	9	10	11

Second Word (DX)

Vector (010)											
Escape	+ −	10-Bit Delta X									
0	1	2	3	4	5	6	7	8	9	10	11

Vector Continue (011)											
Intensify	+ −	10-Bit Delta Y									
0	1	2	3	4	5	6	7	8	9	10	11
Escape		10-Bit Delta X									
0	1	2	3	4	5	6	7	8	9	10	11

Short Vector (100)											
Intensify	+ −	Delta Y				Escape	+ −	Delta X			
0	1	2	3	4	5	6	7	8	9	10	11

Six-bit format

Character (101)											
Character 1						Character 2					
0	1	2	3	4	5	6	7	8	9	10	11

Seven-bit format

Ignored					Character						
0	1	2	3	4	5	6	7	8	9	10	11

Grophplot (110)											
Escape	Set Y Set X	X or Y Coordinate									
0	1	2	3	4	5	6	7	8	9	10	11

MNEMONIC SUMMARY

Mnemonic Symbol	Octal Code	Operation
LPOF	0040	Light pen off.
LPON	0060	Light pen on.
SC1	0400	Set scole to X1.
SC2	0500	Set scale to X2.
SC4	0600	Set scale to X4.
SC8	0700	Set scale to X8.
INT*	0010	Set the intensity.
EDS	1001	Enter data stote.
CCB	1002	Clear coordinate bits.
CSB	1004	Cleor sector bits.
POINT	1100	Set mode to 0.
INCR	1110	Set mode to 1.
VEC	1120	Set mode to 2.
VECON	1130	Set mode to 3.
SVEC	1140	Set mode to 4.
CHAR	1150	Set mode to 5.
GRAPH	1160	Set mode to 6.
CLDF	1200	Cleor flog.
STOP	1400	Stop disploy.
JUMP	2000	Jump to 15-bit oddress contained in last digit ond the next word addressed.
PJMP	2010	Jump to subroutine addressed the same as JUMP.
LPOF	0040	Light pen off.
LPON	0060	Light pen on.
SC1	0400	Set scole to X1.
SC2	0500	Set scale to X2.
SC4	0600	Set scale to X4.
SC8	0700	Set scole to X8.
POP	3000	Exit from subroutine to next oddress ofter PJMP.
PEDS	3001	Pop and enter doto stote.
PNI	3002	Pop ond inhibit restoring intensity.

MNEMONIC SUMMARY (continued)

Mnemonic Symbol	Octal Code	Operation
PNLS	3004	Pop and inhibit restoring light pen and scale.
PNM	3010	Pop and inhibit restoring mode.
LPOF	0040	Light pen off.
LPON	0060	Light pen on.
SC1	0400	Set scale to X1.
SC2	0500	Set scale to X2.
SC4	0600	Set scale to X4.
SC8	0700	Set scale to X8.
SK1	4000	Skip if any of the selected buttons are 0.
INV	0400	Invert sense of test (skip if any selected button is 1).
CLAT	0200	Clear buttons tested after test.
COAT	0100	Complement buttons tested after test.
SK2	5000	Skip if any of the selected buttons are 0.
INV	0400	Invert sense of test (skip if any of the selected buttons are 1).
CLAT	0200	Clear buttons tested after test.
COAT	0100	Complement buttons tested after test.
SK3	6000	Arithmetically compare pushbuttons (0-5) with last two digits of instruction; skip if not equal.
SK4	6100	Same as SK3 but for buttons 6-11.
SKIP	6240	Unconditional skip (two locations).
SNSZ	6220	Skip if sector 0 flag is not up.
SPB1	6210	Skip if push button (0-5) flag is down.
SPB2	6204	Skip if push button (6-11) flag is down.
SCUP	6340	Count scale up.
SCDN	6360	Count scale down.
INTUP	6310	Count intensity up.
INTDN	6314	Count intensity down.
BKON	6302	Blink on.
BKOF	6301	Blink off.
SG0	6400	Set slave group 0.
SG1	6500	Set slave group 1.

MNEMONIC SUMMARY (continued)

Mnemonic Symbol	Octal Code	Operation
SG2	6600	Set slave group 2.
SG3	6700	Set slave group 3.
SU0	0040	Turn light pen and intensity off on unit 0.
LP0	0060	Unit 0 light pen on.
IT0	0050	Unit 0 intensity on.
SU1	0004	Turn light pen and intensity off on unit 1.
LP1	0006	Unit 1 light pen on.
IT1	0005	Unit 1 intensity on.

IOT SUMMARY

IOT	Octal Code	Meaning	Page
RPDP	6051	Read Push Down Pointer	29
RXP	6052	Read x Position Register	29
RYP	6054	Read y Position Register	29
RDAC	6061	Read Display Address Counter	29
RS1	6062	Read Status 1	29
RS2	6064	Read Status 2	30
RPB	6071	Read Push Buttons	31
RSG1	6072	Read Slave Group 1	31
RSG2	6074	Read Slave Group 2	31
RCG	6304	Read Character Generator	31
SPDP	6135	Set the Push Down Pointer	32
SIC	6145	Set Initial Conditions	32
LBF	6155	Load Break Field	33
SCG	6303	Set Character Generator	34
INIT	6165	Initialize the Display	34
RES1	6174	Resume After Light Pen Hit, Edge, or External Stop Flag	36
RES2	6164	Resume After Stop Code	36
CFD	6161	Clear Display Flags	36
STPD	6154	Stop Display (External)	36
SPLP	6132	Skip on Light Pen Hit Flag	36
SPSP	6142	Skip on Slave Light Pen Hit Flag	36
SPES	6151	Skip on External Stop Flag	36
SPEF	6152	Skip on Edge Flag	36
SPSF	6171	Skip on Internal Stop Flag	37
SPM1	6172	Skip on Manual Interrupt	37

RS1

L.P. Hit Flag	Vertical Edge Flag	Horizontal Edge Flag	Internal Stop	Sector Zero	Control State	Manual Interrupt	P.B. Hit	Display Interrupt	Break Field		
0	1	2	3	4	5	6	7	8	9	10	11

RS2

Byte	L.P. Enable	Y Position	X Position	Scale		Mode			Intensity		
0	1	2	3	4	5	6	7	8	9	10	11

RCG

Character	CB	Spare	Case	CHSZ	Spare	SAR					
0	1	2	3	4	5	6	7	8	9	10	11

SIC

Edge Interrupt	L.P. Interrupt	L.P. Resume Options		Y Dimension		X Dimension		Intensify All Points	Inhibit Edge Flags	P.B. Interrupt	Internal Stop Interrupt
0	1	2	3	4	5	6	7	8	9	10	11

SCG

Spare			Case	CHSZ	Spare	SAR					
0	1	2	3	4	5	6	7	8	9	10	11

Form AEC-427
(6/68)
AECM 3201

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO.

COO-1469-0142

2. TITLE

DESIGN OF A DISPLAY PROCESSING UNIT IN A MULTI-TERMINAL ENVIRONMENT

3. TYPE OF DOCUMENT (Check one):

- [X] a. Scientific and technical report
- [ ] b. Conference paper not to be published in a journal:
  - Title of conference _____
  - Date of conference _____
  - Exact location of conference _____
  - Sponsoring organization _____
- [ ] c. Other (Specify) _____

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- [X] a. AEC's normal announcement and distribution procedures may be followed.
- [ ] b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- [ ] c. Make no announcement or distrubution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear, Professor
and Principle Investigator

Organization

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Signature Charles W Gear

Date July 30, 1969

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

8. PATENT CLEARANCE:

- [ ] a. AEC patent clearance has been granted by responsible AEC patent group.
- [ ] b. Report has been sent to responsible AEC patent group for clearance.
- [ ] c. Patent clearance not required.